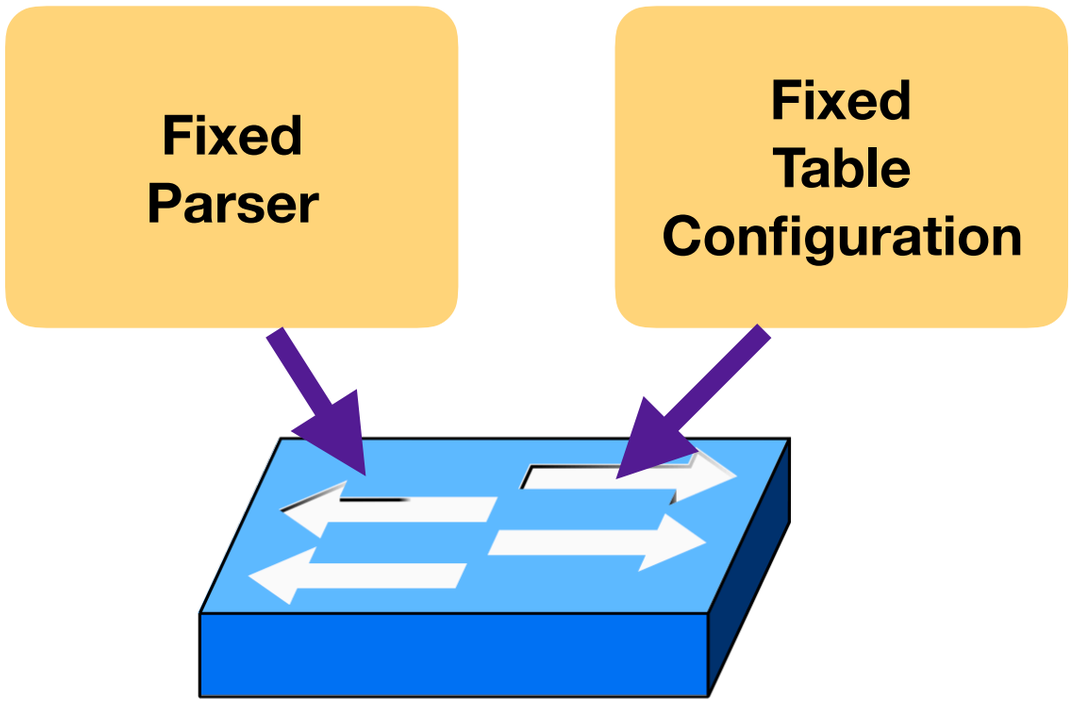
Elastic Switch Programming with P4All

Mary Hogan*, Shir Landau-Feibish*, Mina Tahmasbi Arashloo+, Jennifer Rexford*, David Walker*, Rob Harrison^

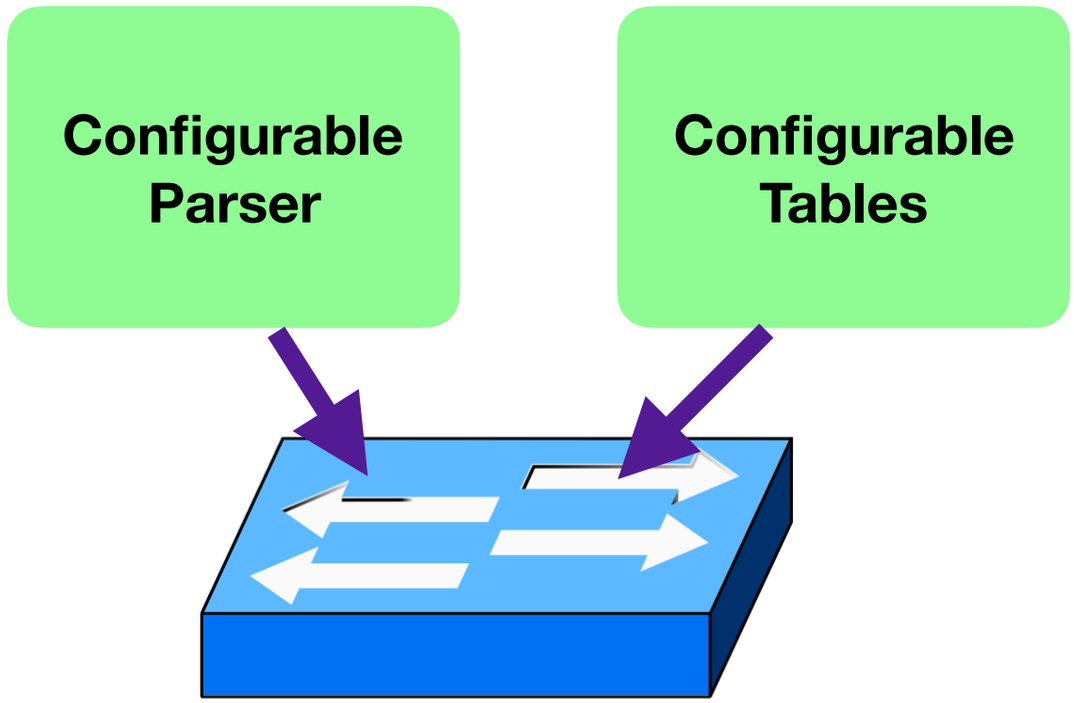
*Princeton University, +Cornell University, ^United States Military Academy West Point

Traditional data plane switches hinder innovation



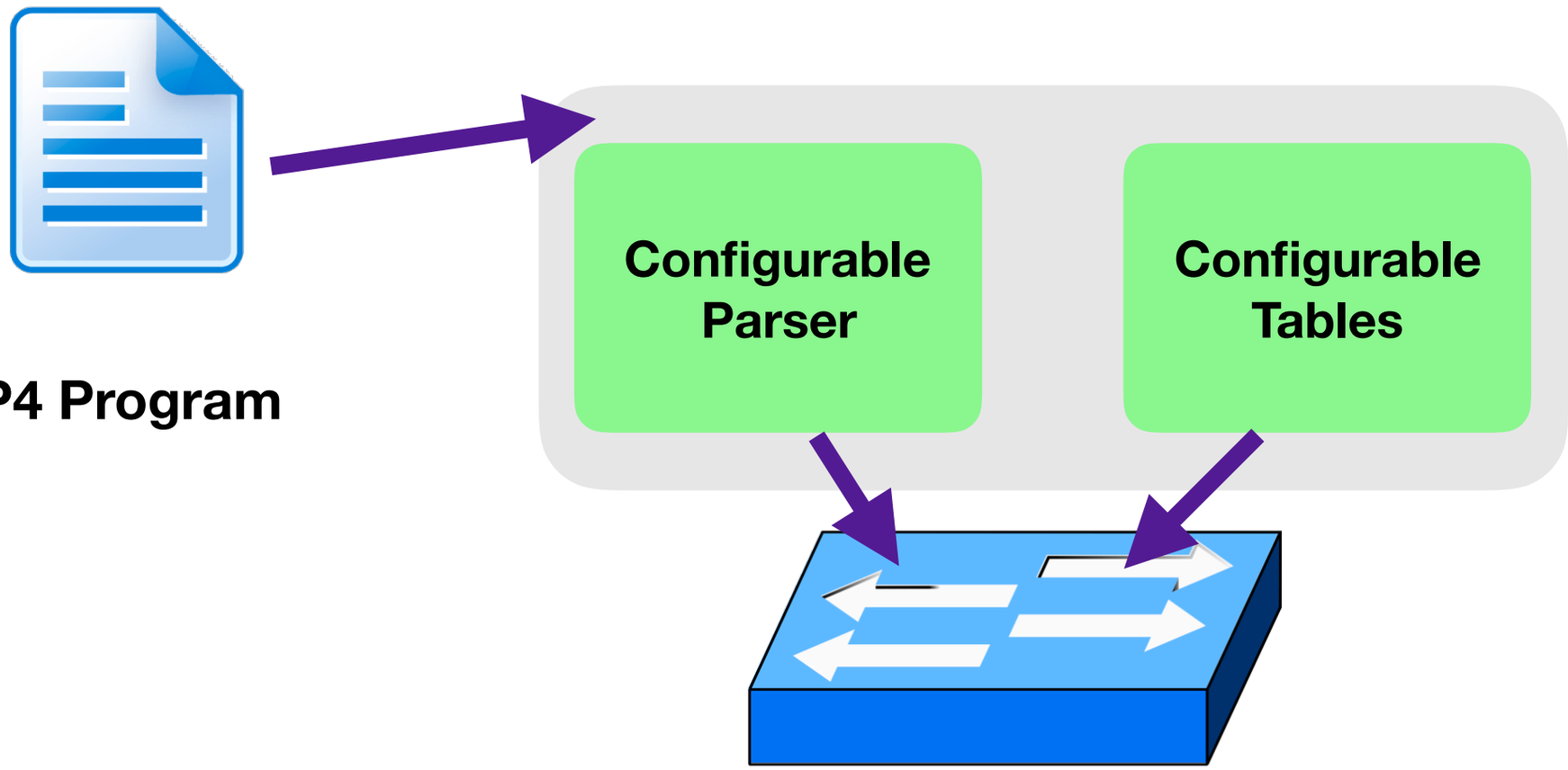
Fixed-function switch

Protocol Independent Switch Architecture



PISA switch

Programming Protocol Independent Packet Processors



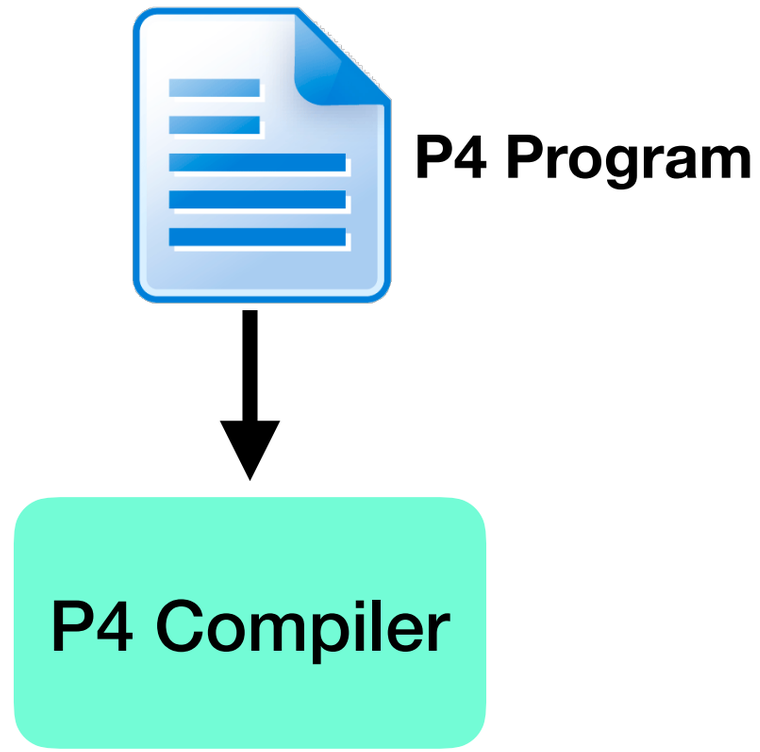
P4 Program

PISA switch

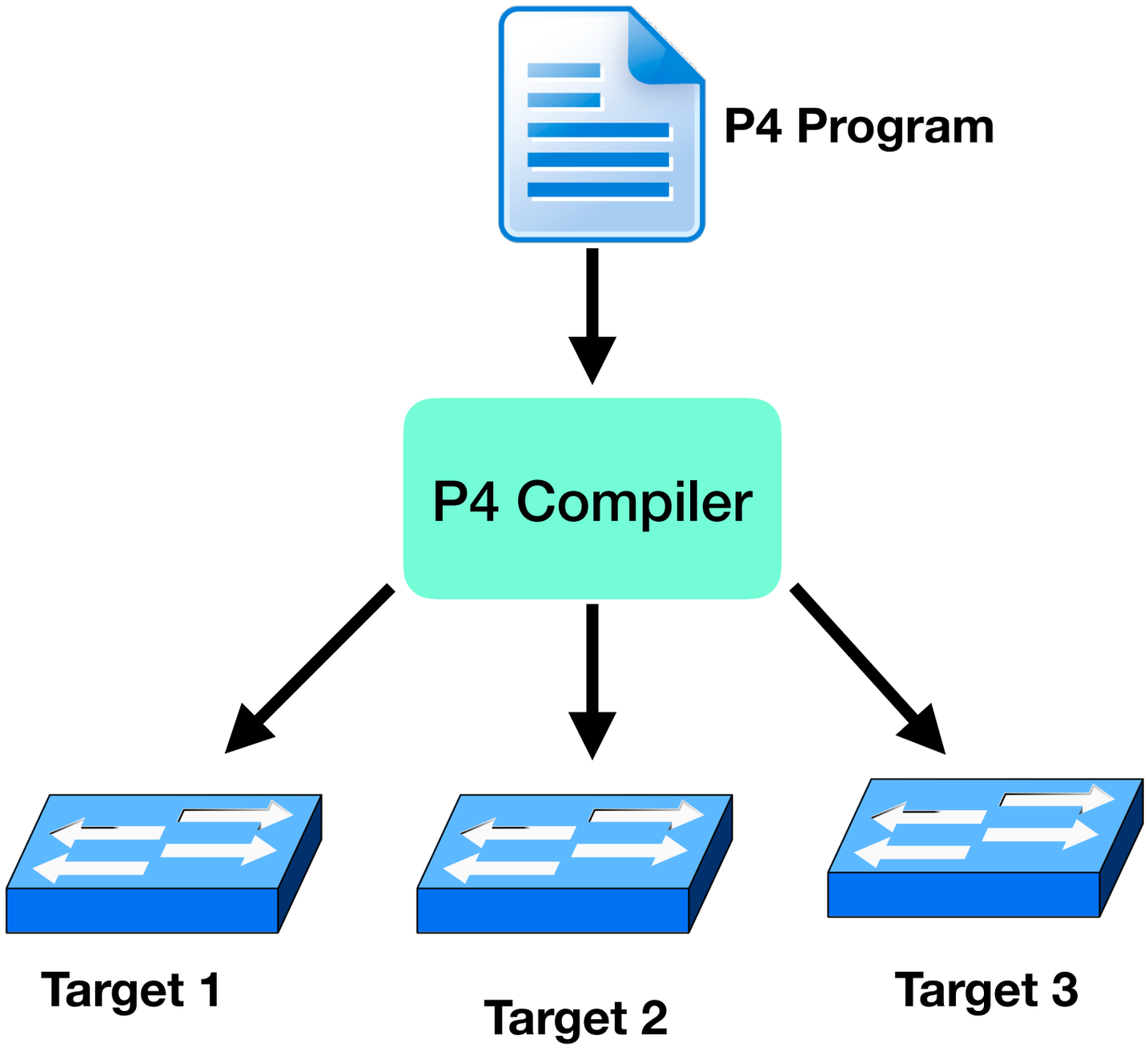
P4 code should be reusable

P4 Program

P4 code should be reusable



P4 code should be reusable



P4 code is **not** reusable

P4 code is **not** reusable

Data structures (e.g., count-min sketch) are valid for a range of sizes

P4 code is **not** reusable

Data structures (e.g., count-min sketch) are valid for a range of sizes

P4 requires explicit definition of size - amount of memory, logical tables, etc.

P4 code is **not** reusable

Data structures (e.g., count-min sketch) are valid for a range of sizes

P4 requires explicit definition of size - amount of memory, logical tables, etc.

P4 data structures must be adapted to fit specific targets

P4 code is **not** reusable

Data structures (e.g., count-min sketch) are valid for a range of sizes

P4 requires explicit definition of size - amount of memory, logical tables, etc.

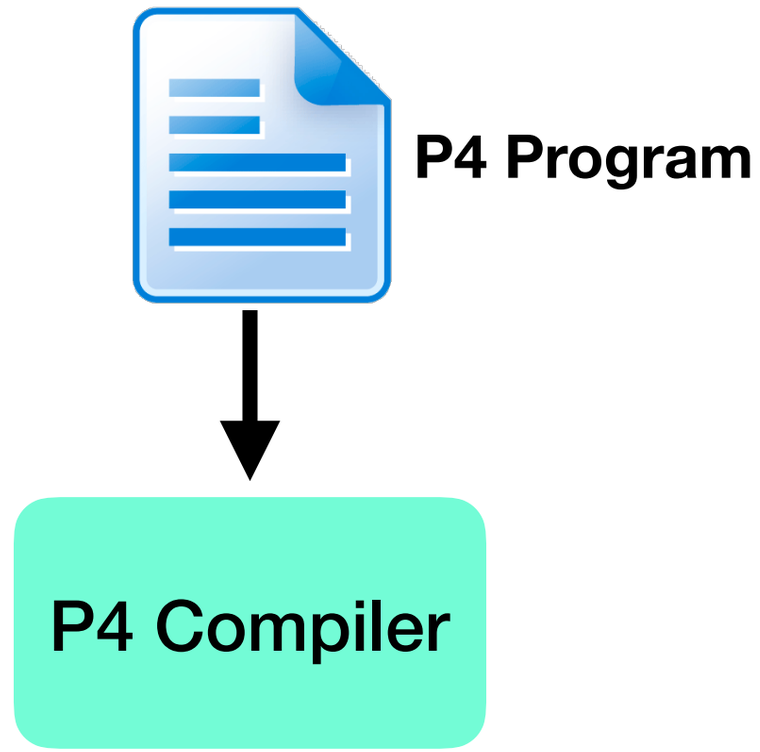
P4 data structures must be adapted to fit specific targets

Commonly used data structures are rewritten often

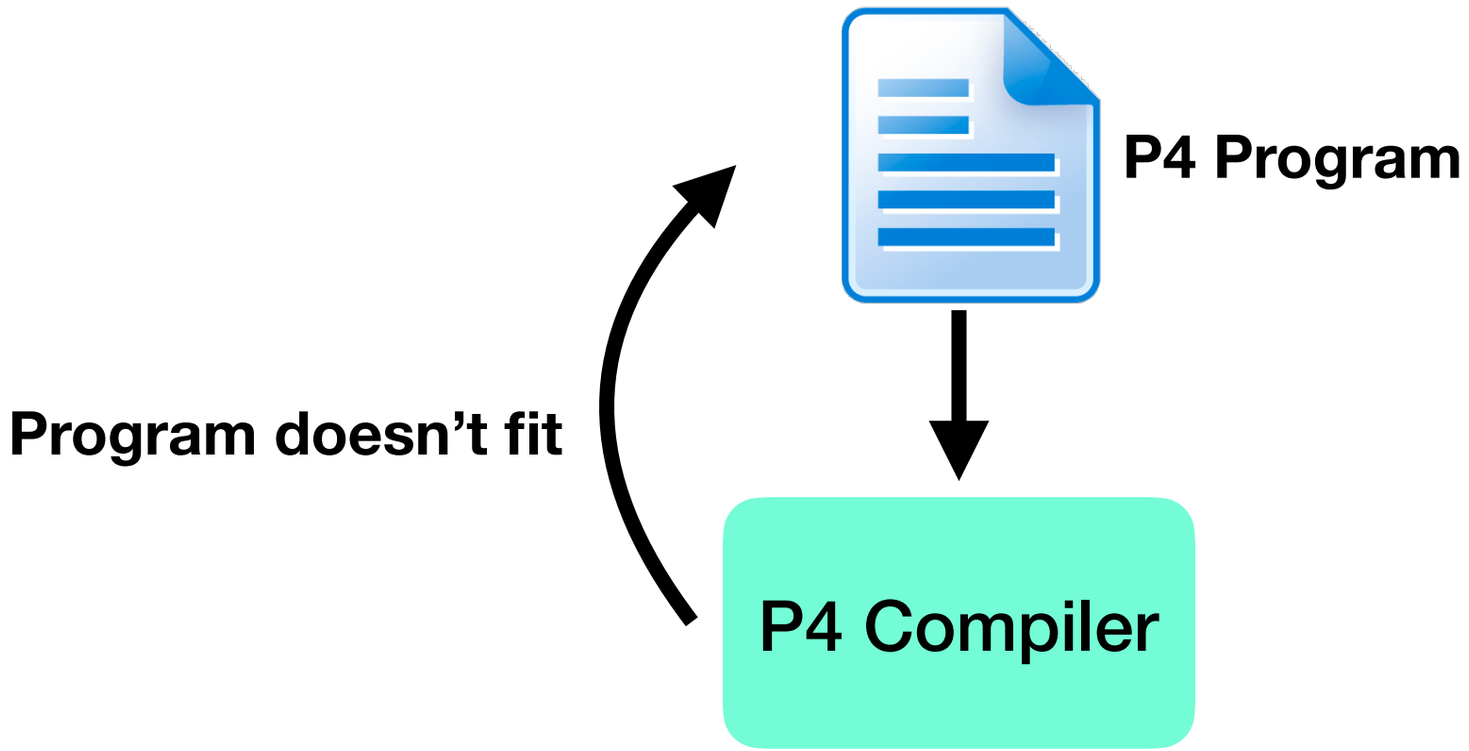
Circular Development

P4 Program

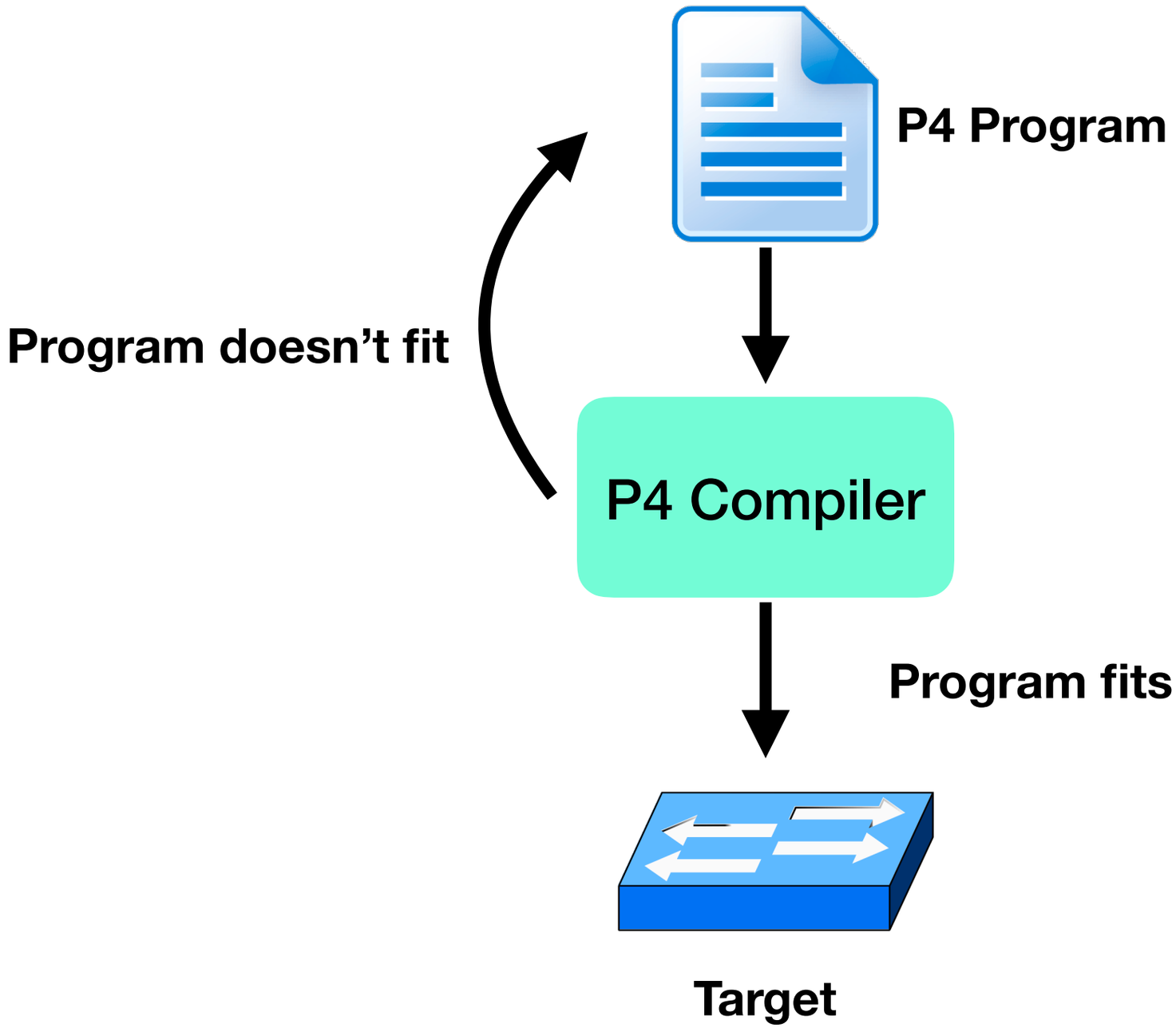
Circular Development



Circular Development



Circular Development



P4All mitigates circularity

P4All mitigates circularity

P4All streamlines development with elastic data structures

P4All mitigates circularity

P4All streamlines development with elastic data structures

Elastic data structures are defined by symbolic values

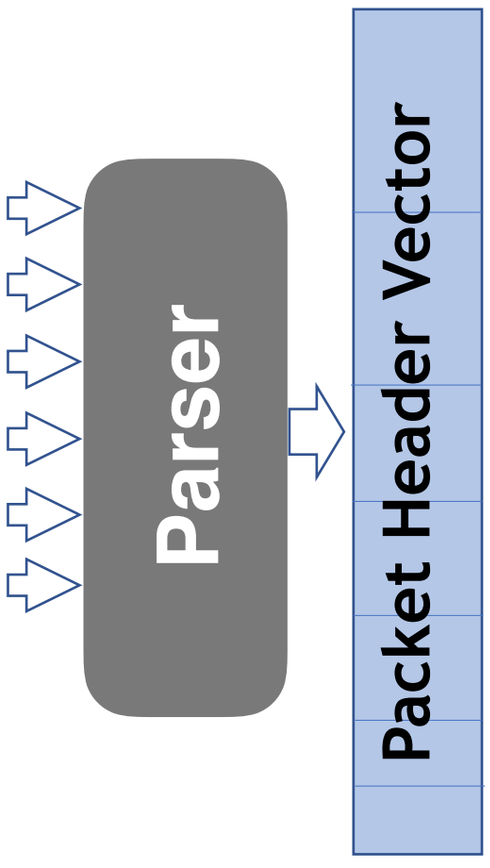
P4All mitigates circularity

P4All streamlines development with elastic data structures

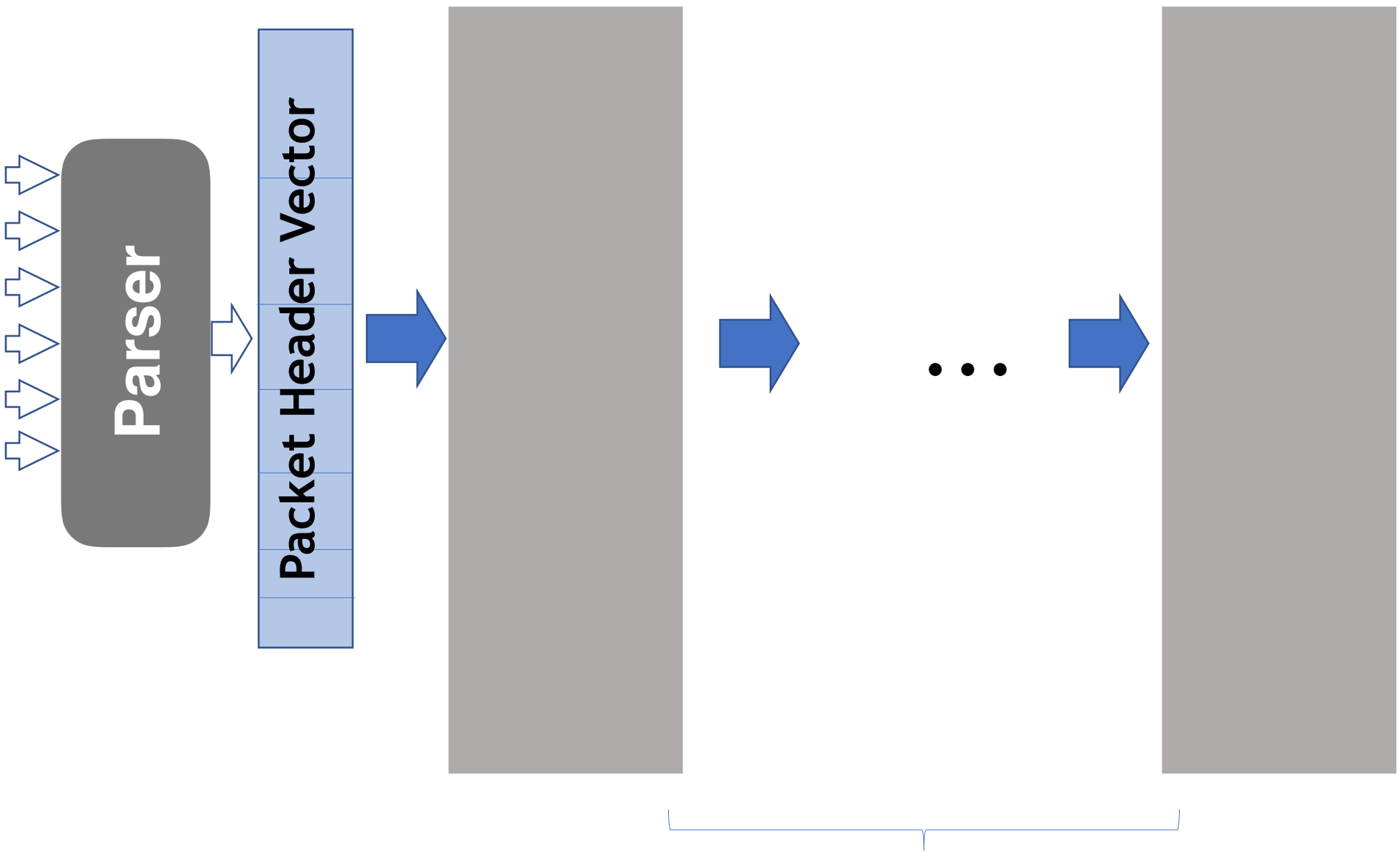
Elastic data structures are defined by symbolic values

A single P4All program optimally compiles without rewriting

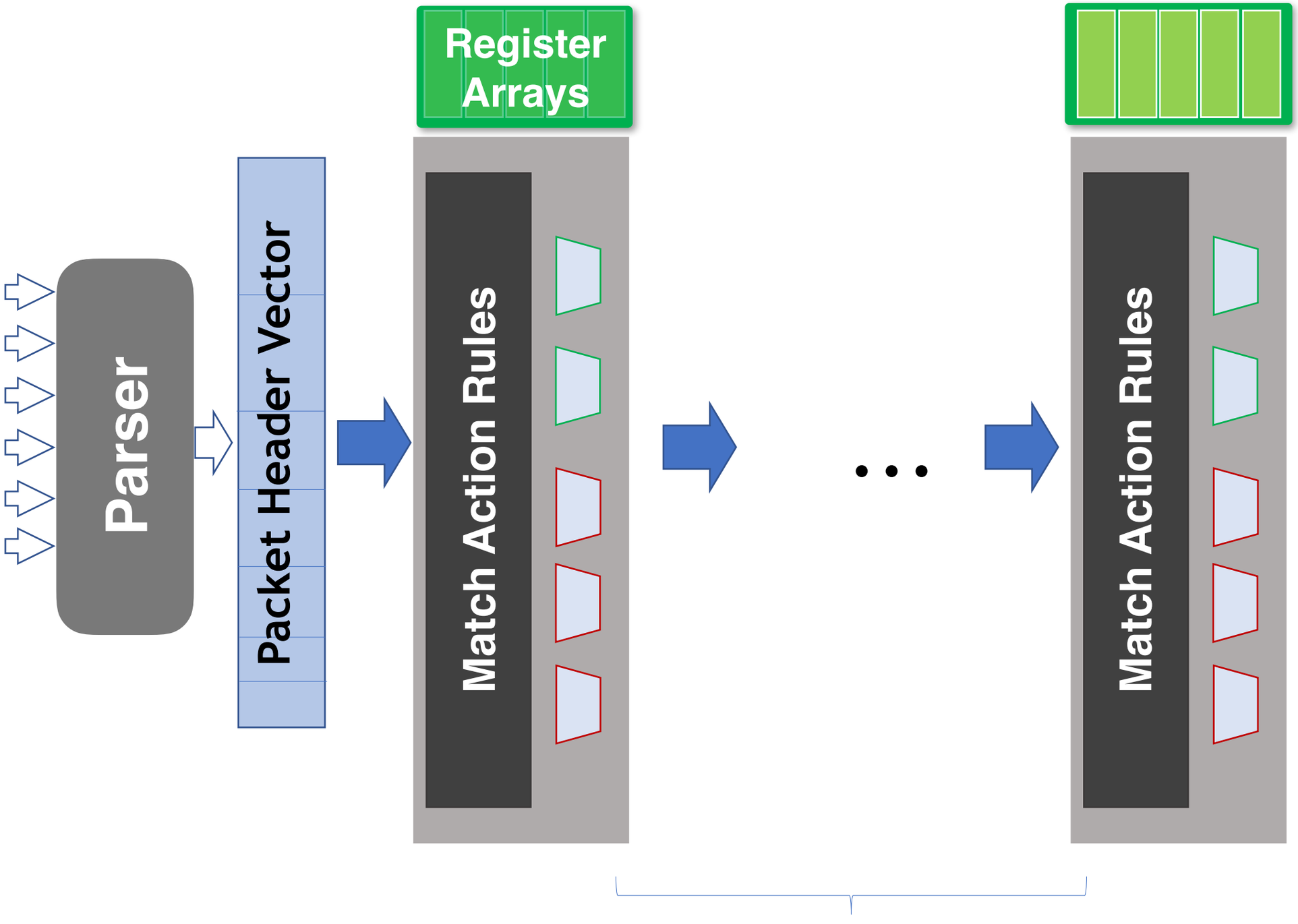
PISA



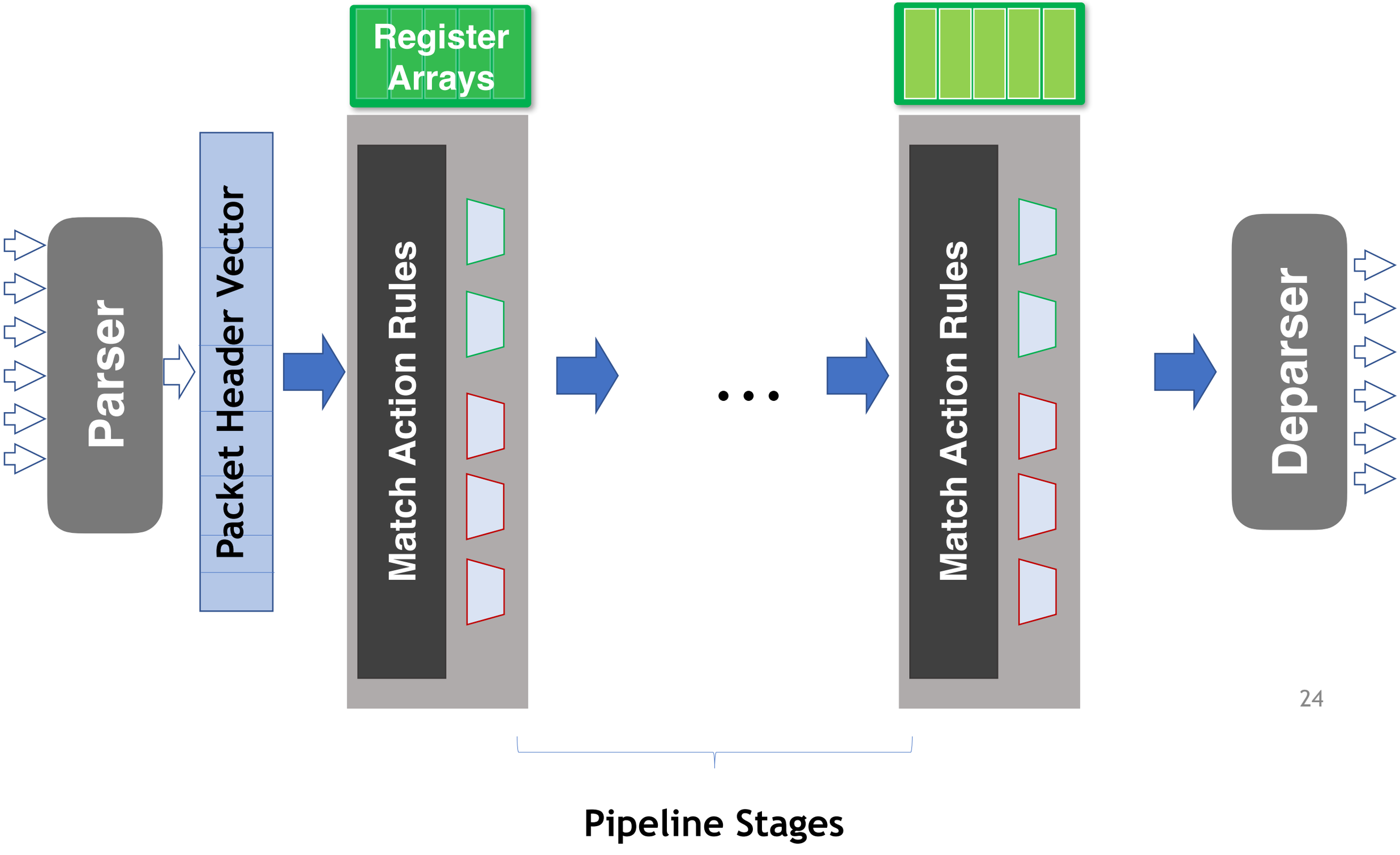
PISA



PISA


Register Arrays
Parser
Packet Header Vector
Match Action Rules
Match Action Rules
Pipeline Stages

PISA


Register Arrays
Register Arrays
Parser
Packet Header Vector
Match Action Rules
Match Action Rules
Deparser
Pipeline Stages

Count-Min Sketch

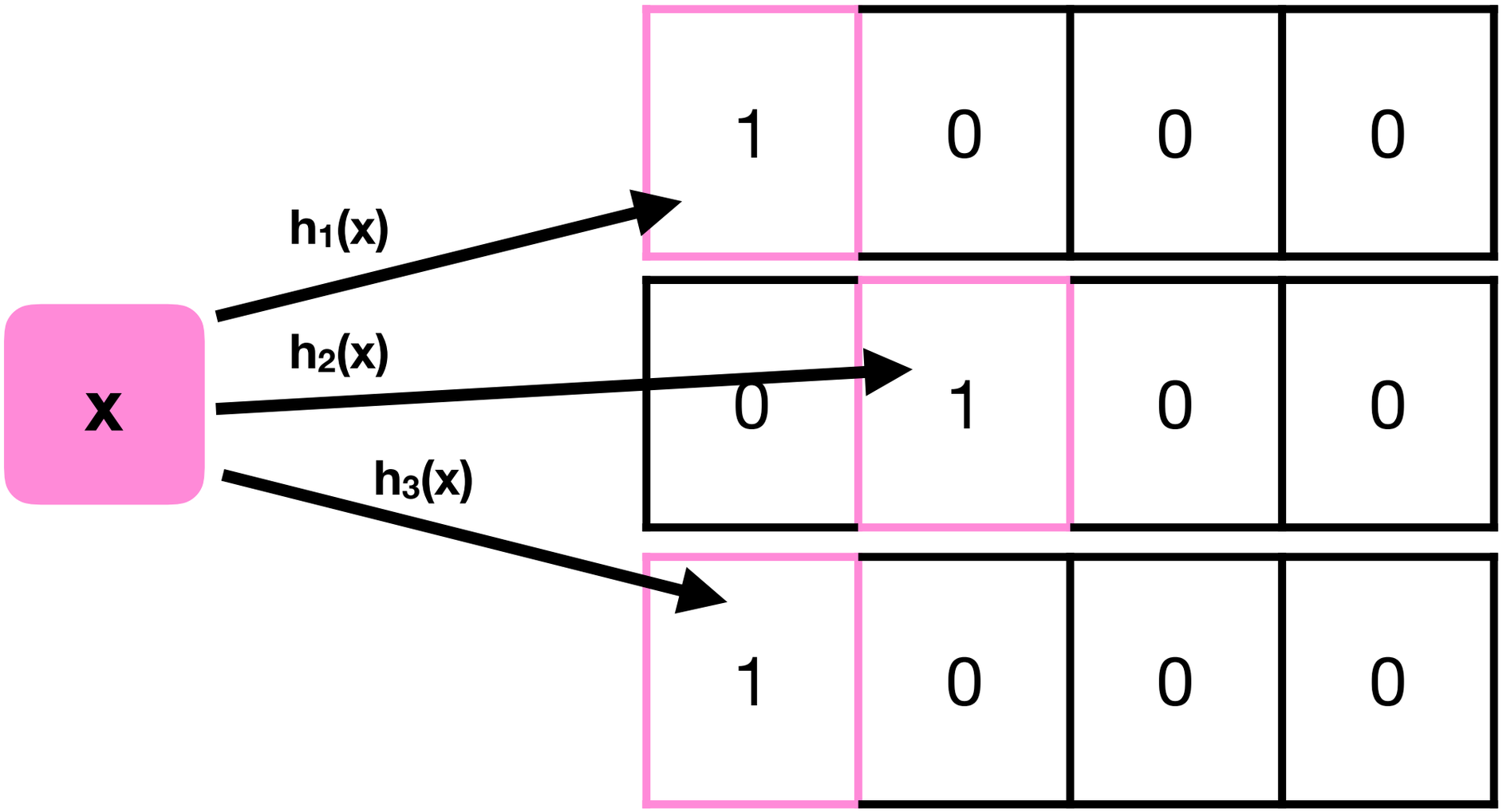
0	0	0	0
0	0	0	0
0	0	0	0

Count-Min Sketch

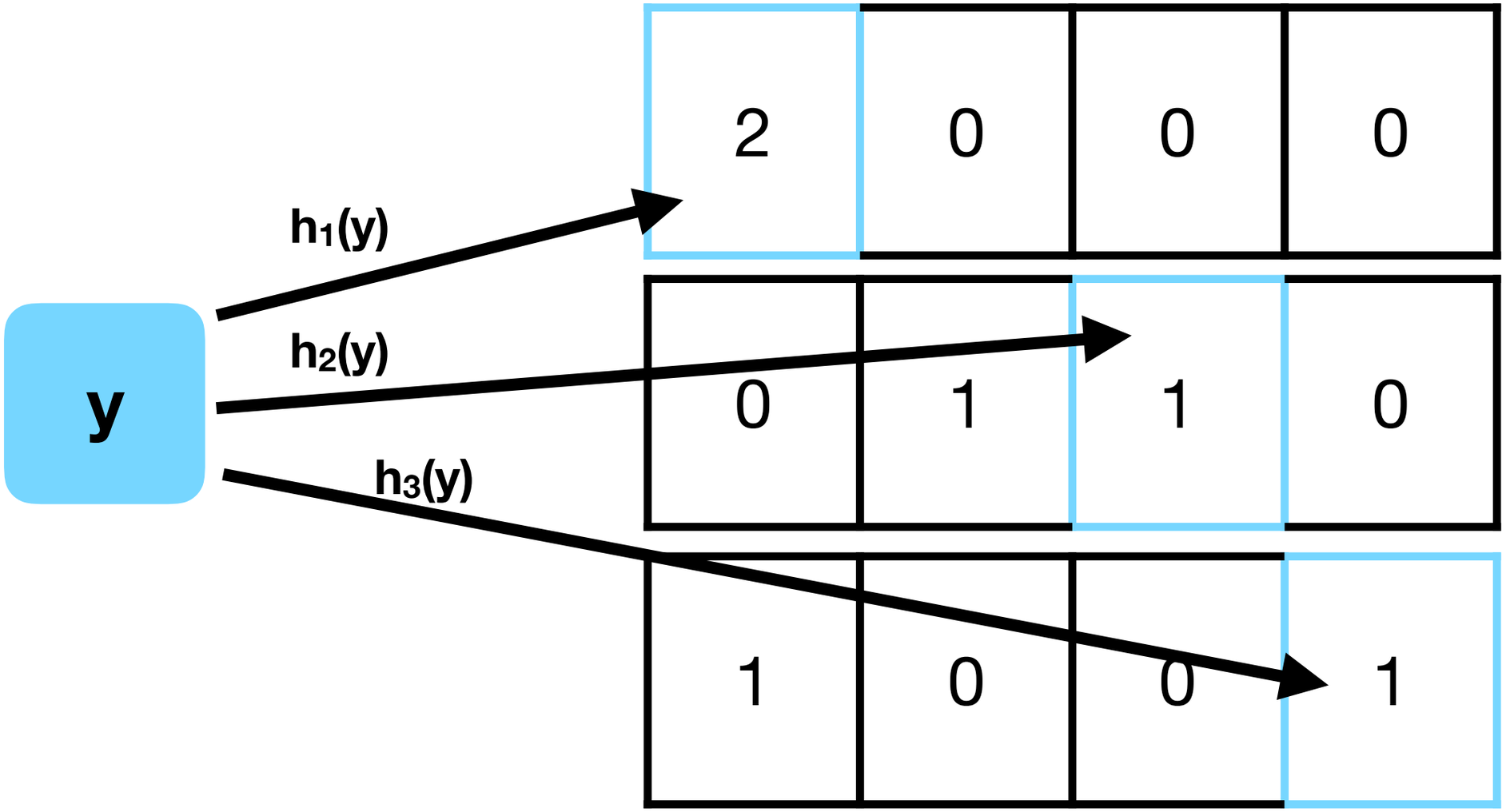
x

0	0	0	0
0	0	0	0
0	0	0	0

Count-Min Sketch



Count-Min Sketch



Count-Min Sketch

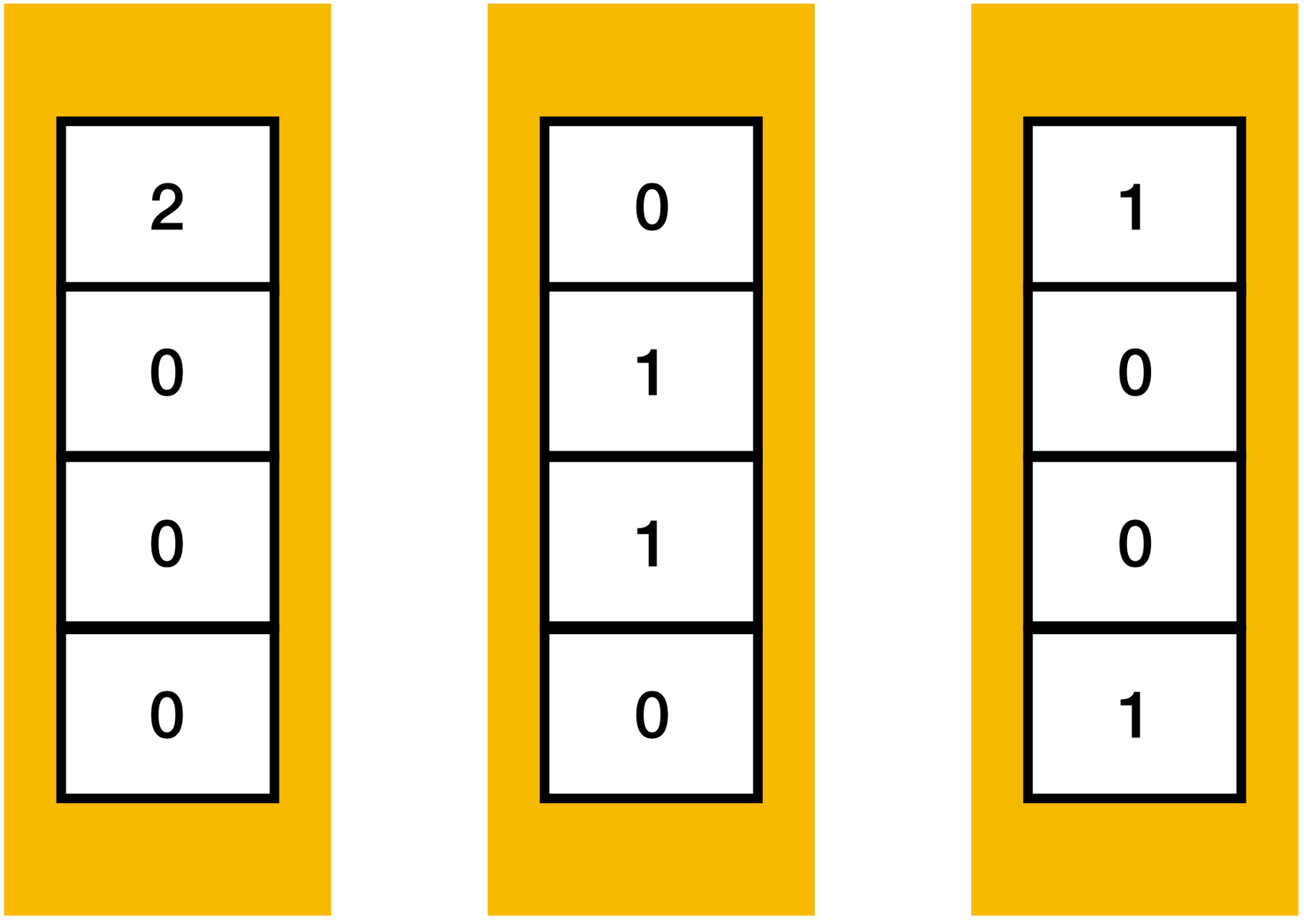
x

2	0	0	0
0	1	1	0
1	0	0	1

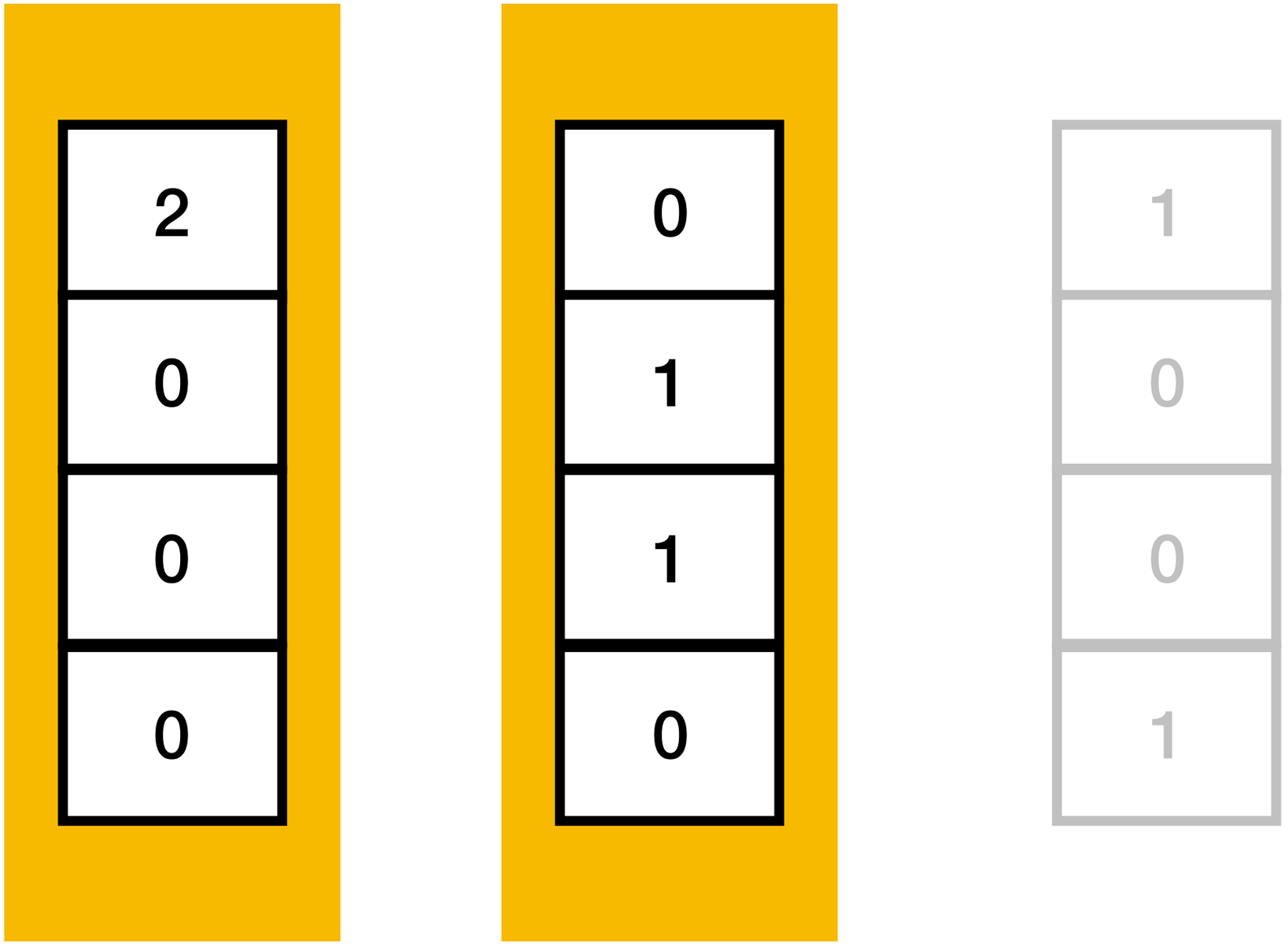
Count(x) = 1

CMS in PISA

CMS in PISA



CMS in PISA



Outline

Data Structures in P4

P4All Constructs

- Elastic Data Structures
- Elastic Operations
- Utility Functions

Ongoing and Future Work

Outline

Data Structures in P4

P4All Constructs

Elastic Data Structures

Elastic Operations

Utility Functions

Ongoing and Future Work

Count-Min Sketch in P4

```
control Ingress () {
   register<bit<32>>(4) counter1;
   register<bit<32>>(4) counter2;
   register<bit<32>>(4) counter3;
}
```

CMS in P4

```
control Ingress () {
    register<bit<32>>(4) counter1;
    register<bit<32>>(4) counter2;
    register<bit<32>>(4) counter3;
}
```

CMS in P4

```
struct custom_metadata_t {
    bit<32> min;
    bit<32> index1;
    bit<32> count1;
    . . .
}
control Ingress (inout custom_metadata_t meta) {
    register<bit<32>>(4) counter1;
    . . .
}
```

CMS in P4

```
struct custom_metadata_t {
    bit<32> min;
    bit<32> index1;
    bit<32> count1;
    . . .
}
control Ingress (inout custom_metadata_t meta) {
    register<bit<32>>(4) counter1;
    . . .
}
```

CMS in P4

```
struct custom_metadata_t {
    bit<32> min;
    bit<32> index1;
    bit<32> count1;
    . . .
}
control Ingress (inout custom_metadata_t meta) {
    register<bit<32>>(4) counter1;
    . . .
}
```

CMS in P4

```
struct custom_metadata_t {
    bit<32> min;
    bit<32> index1;
    bit<32> count1;
    . . .
}
```

```
control Ingress (inout custom_metadata_t meta) {
    register<bit<32>>(4) counter1;
    action incr_1() {
        /* action to update row 1 */
    }
    action min_1() {
        /* action to set global min */
    }
}
```

CMS in P4

```
struct custom_metadata_t {
    bit<32> min;
    bit<32> index1;
    bit<32> count1;
    . . .
}
```

```
control Ingress (inout custom_metadata_t meta) {
    register<bit<32>>(4) counter1;
    action incr_1() {
        /* action to update row 1 */
    }
    action min_1() {
        /* action to set global min */
    }
}
```

CMS in P4

```
struct custom_metadata_t {
    bit<32> min;
    bit<32> index1;
    bit<32> count1;
    . . .
}

control Ingress (inout custom_metadata_t meta) {
    . . .
    apply {
        incr_1();
        if (meta.count1 < meta.min) {
            min_1();    }
        . . .
    }
}
```

Outline

Data Structures in P4

P4All Constructs

Elastic Data Structures

Elastic Operations

Utility Functions

Ongoing and Future Work

Elastic CMS

```
control Ingress (…) {
   register<bit<32>>(4) counter1;
   register<bit<32>>(4) counter2;
   register<bit<32>>(4) counter3;
   . . .
   }
```

P4

```
symbolic int rows;
symbolic int cols;
assume (0 <= rows) && (rows < 8);
```

P4All

Elastic CMS

```
control Ingress (…) {
   register<bit<32>>(4) counter1;
   register<bit<32>>(4) counter2;
   register<bit<32>>(4) counter3;
   . . .
   }
```

P4

```
symbolic int rows;
symbolic int cols;
assume (0 <= rows) && (rows < 8);
control Ingress (…) {
   register<bit<32>>(cols)[rows] counters;
}
```

P4All

Elastic CMS

```
control Ingress (…) {
   register<bit<32>>(4) counter1;
   register<bit<32>>(4) counter2;
   register<bit<32>>(4) counter3;
   . . .
   }
```

P4

```
symbolic int rows;
symbolic int cols;
assume (0 <= rows) && (rows < 8);
control Ingress (…) {
   register<bit<32>>(cols)[rows] counters;
}
```

P4All

Elastic CMS

```
struct custom_metadata_t {
   bit<32> min;
   bit<32> index1;
   bit<32> count1;
   . . .
}
```

P4

```
symbolic int rows;
symbolic int cols;
assume (0 <= rows) && (rows < 8);

struct custom_metadata_t {
   bit<32> min;
   bit<32>[rows] indices;
   bit<32>[rows] counts;
}
```

P4All

Outline

Data Structures in P4

P4All Constructs

Elastic Data Structures

Elastic Operations

Utility Functions

Ongoing and Future Work

Elastic Operations

```
control Ingress (. . .) {
   action incr_1() { . . .}
   . . .
}
```

P4

```
symbolic int rows;
symbolic int cols;
assume (0 <= rows) && (rows < 8);

control Ingress (. . .) {
   register<bit<32>>(cols)[rows] counters;
   action incr()[int index] { . . .}
}
```

P4All

Elastic Operations

```
control Ingress (. . .) {
   action incr_1() {. . .}
   . . .
   apply {
      incr_1();
      . . .
   }
}
```

P4

```
symbolic int rows;
symbolic int cols;
assume (0 <= rows) && (rows < 8);

control Ingress ( . . .) {
   action incr()[int index] { . . . }
   apply {
      for (i < rows) {
         incr()[i];
      }
   }
}
```

P4All

Elastic Operations

```
control Ingress (. . .) {
   action incr_1() {. . .}
   . . .
   apply {
      incr_1();
      . . .
   }
}
```

P4

```
symbolic int rows;
symbolic int cols;
assume (0 <= rows) && (rows < 8);

control Ingress ( . . .) {
   action incr()[int index] { . . . }
   apply {
      for (i < rows) {
         incr()[i];
      }
   }
}
```

P4All

Outline

Data Structures in P4

P4All Constructs

Elastic Data Structures

Elastic Operations

Utility Functions

Ongoing and Future Work

Utility Functions

Utility Functions

ϵ

CMS error

Utility Functions

$$\texttt{cols} = 3\,(1/\epsilon)^{1/\alpha}$$

Symbolic CMS columns

CMS error

Utility Functions

$$\texttt{cols} = 3\,(1/\epsilon)^{1/\alpha}$$

Symbolic CMS columns

CMS error

Workload-dependent parameter

Utility Functions

```
symbolic int rows;
symbolic int cols;
control Ingress (…) {
    …
}

optimize cms_error;
```

Outline

Data Structures in P4

P4All Constructs

Elastic Data Structures

Elastic Operations

Utility Functions

Ongoing and Future Work

Ongoing + Future work

Design representative utility functions

Ongoing + Future work

Design representative utility functions

Support dynamic utility functions

Ongoing + Future work

Design representative utility functions

Support dynamic utility functions

Object-oriented programming model

Elastic Switch Programming with P4All

Mary Hogan, Shir Landau-Feibish, Mina Tahmasbi Arashloo, Jennifer Rexford, David Walker, Rob Harrison